graphics administration guide
for HP-UX 11.X (IPF version)

Document Part Number: 5969-3151

June 2003

WARNING: Text set off in this manner indicates that failure to follow directions could result in bodily harm or loss of life.

CAUTION: Text set off in this manner indicates that failure to follow directions could result in damage to equipment or loss of information.

graphics administration guide
for HP-UX 11.X (IPF version)
First Edition (June 2003)

Document Part Number: 5969-3151

Contents

X Server configuration details

miscellaneous topics

1

preface

Documentation for each graphical application is provided by two manuals: one specific to the application and this manual which provides information common to all of the applications. For example, the primary user interface of the workstation is X Windows which is required whether or not you use any 3D APIs. Because each API interacts with the X Server in the same way, X Server information is provided in this guide.

document conventions

This table lists the typographical conventions used in this document:

Typographical Conventions

Typographical Convention	Meaning
`mknod /usr/include`	Verbatim computer literals are in computer font. Text in this style is letter-for-letter verbatim and, depending on the context, should be typed in exactly as specified, or is named exactly as specified.
. . .the *<device_id. . .>*	Conceptual values are in italic type, enclosed in angle brackets. These items are not verbatim values, but are descriptors of the type of item it is, and the user should replace the conceptual item with whatever value is appropriate for the context.
In every case...	Emphasized words are in italics.
. . .device is a **freen**. . .	New terms being introduced are in bold-faced type.

2

path names

This chapter contains information on locating files that reside in the file system.

finding files with “whence” and “whereis”

There are two main methods of finding commands if you know the name of the command you’re looking for. The first method is to use the Korn shell command `whence`, which tells you where a command resides. If you’re not using the Korn shell, you can use the system command `whereis`. To use whence, enter:

```
$ whence mknod
```

The path for the command you’re looking for is returned:

```
/etc/mknod
```

The above approach has two limitations:

- Both `whence` and `whereis` only find executable files; that is, commands (both compiled programs and shell scripts). `whence` and `whereis` do not find non-executable files, even if they are in your PATH. To find nonexecutable files, use `find`, which is described in the next section.
- The directory in which the command resides must be one of the entries in the PATH variable; if it is not, it won’t be found. So in a sense, `whence` and `whereis` can only find things if you tell them where to look. They are valuable when a command is in your PATH but you do not remember where it is. Also, if you have two commands of the same name in two different directories, `whence` and `whereis` tell you which one will be found first, and thus executed.

finding files with “find”

The `find` command finds any file in your file system, executable or not. For example, to locate an include file, you would execute:

```
$ find / -name '<file_name>'
```

where <*file_name*> is the name of the file you’re looking for. In this example, “/” is the root directory. If you specify the correct file name, and it is somewhere in the file system, the `find` command will find it, though it make take a while. You can shorten the search time by including a subdirectory. For example:

```
find /opt -name '<file_name>'
```

searches only the `/opt` directory.

You also can specify a partial filename. The `find` command locates all files that contain a specified substring in their names.

`find` has many other options for refining a search; see the man page for details.

path names

/opt/graphics/OpenGL files

`contrib/libglut`	Utilities found in the OpenGL Utility Toolkit as mentioned in the *OpenGL Programming for the X Window System* manual
`contrib/libwidget`	Motif widget library and source code
`contrib/xglinfo`	Utility to print display and visual information for OpenGL with the X Window system
`demos`	Sample OpenGL programs, including source code
`doc`	OpenGL documentation including reference pages
`include/GL`	Header files needed for OpenGL development
`lbin`	Run-time executables
`lib/hpux32`	32-bit run-time shared libraries
`lib/hpux64`	64-bit run-time shared libraries

/usr/lib files

`hpux32/X11/Xserver/modules/xf86/extensions`	32-bit extension libraries needed to run OpenGL with the X Server
`hpux64/X11/Xserver/modules/xf86/extensions`	64-bit extension libraries needed to run OpenGL with the X Server

/opt/graphics/common/lib files

`hpux32`	32-bit run-time shared libraries needed to run OpenGL
`hpux64`	64-bit run-time shared libraries needed to run OpenGL

3

compiling your application

This chapter provides information for compiling your application using the OpenGL application programming interface (API). Compiling examples are given for C.

compiling OpenGL applications

To compile a program that does not use the OpenGL utilities, use a makefile that looks like this:

```
INCDIR=-I/opt/graphics/OpenGL/include
LIBDIR32=-L/opt/graphics/OpenGL/lib/hpux32
LIBDIR64=-L/opt/graphics/OpenGL/lib/hpux64
LIBS=-lGL -lXext -lX11 -lm -ldld
prog32: prog.c
    cc -Ae $(INCDIR) $(LIBDIR32) -o prog32 prog.c $(LIBS)
prog64: prog.c
     cc -Ae +DD64 $(INCDIR) $(LIBDIR64) -o prog64 prog.c
$(LIBS)
```

To compile a program that does use the OpenGL utilities, use a makefile that looks like this:

```
INCDIR= -I/opt/graphics/OpenGL/include
LIBDIR32=-L/opt/graphics/OpenGL/lib/hpux32
LIBDIR64=-L/opt/graphics/OpenGL/lib/hpux64
LIBS=-lGLU -lGL -lXext -lX11 -lm -ldld
prog32: prog.c
  cc -Ae $(INCDIR) $(LIBDIR32) -o prog32 prog.c $(LIBS)
prog64: prog.c
  cc -Ae +DD64 $(INCDIR) $(LIBDIR64) -o prog64 prog.c $(LIBS)
```

kernel graphics drivers

All OEM graphics devices on HP-UX IPF systems use multiple kernel graphics drivers. These kernel drivers include GVID (General VIDeo driver) and separate graphics DLKM (Dynamic Loadable Kernel Modules). All DLKMs are loaded when needed, instead of being statically linked into the kernel. This means that a new kernel does not need to be built every time a new graphics module is needed. GVID is not a DLKM, but it is statically built into the kernel. GVID is device-independent.

The usage of DLKMs is very different for the 11.22 version of HP-UX and subsequent versions of HP-UX. Both versions of DLKMs are discussed in this section.

Like any other part of the system, there is the potential that the DLKMs become corrupted. There are simple commands that allow you to determine if the DLKMs are functioning properly.

correctly configured 11.22 system

To determine if the DLKMs are loaded correctly on HP-UX 11.22 systems, run the command `/usr/sbin/kmadmin -s` (as root). This displays the currently loaded/used DLKMs. The DLKMs that you should see for a correctly configured graphics system are gvid_info, the DRM, and the HIM (Hardware Init Module). For example, for an ATI FireGL X1 device, you should verify that the following three modules are present (there may be other modules listed, but these three are needed):

required DLKMs

Name	ID	Status	Type	Phase
gvid_info	1	LOADED	WSIO	AUTO
drmfglrx	2	UNLOADED	Misc	AUTO
gvid_him_rad	3	UNLOADED	Misc	AUTO

The status of the drmfglrx and gvid_him_rad modules may be LOADED if you are currently using the graphics device (for example, the X Server is running). If you do not see the above modules listed, you may have to reload the modules.

correctly configured 11.23 and later system

To determine if the DLKMs are loaded correctly on HP-UX 11.23 and later systems, run the command `/usr/sbin/kcmodule -v -q <DLKM Name>` (as root). This displays the status of the specified DLKM. The DLKMs that you should look for in a correctly configured graphics system included: gvid_info, the DRM, and the HRM. For example, for an ATI FireGL X1 device, you should verify the following three modules are present: gvid_info, drmfglrx, and gvid_him_rad. The output for the `kcmodule` command with each of these DLKMs would look something like this:

```
Name              drmfglrx
Description       FireGL X1 3D DLKM
Version           0.1.0
State             auto
Cause             explicit
State at Next     auto
Boot
Cause for Next    explicit
Boot
Capable           unused loaded auto unloadable
Depends On        interface HPUX_11_23:1.0.0
```

The status of the drmfglrx/gvid_him_rad modules may be loaded if you are currently using the graphics device (for example, the X Server is running). If the `kcmodule` command returns an error that it cannot find the specified DLKM with a message similar to:

```
Error: There are no modules matching the name you specified.
```

You may have to reload the modules. To see the complete name of all of the possible DLKMs, see the following paragraphs.

To quickly determine if gvid is loaded and running correctly, run the `cat` command on the /dev/gvid_info device file. This returns all of the valid gvid device files (for example, /dev/gvid0, /dev/gvid1, and so forth).

loading a graphics DLKM on HP-UX 11.22

All graphics DLKMs are loaded using the same commands. You must change into the directory where the DLKM resides. This directory should contain three files:

`mod.o:`	The actual DLKM
`master:`	The master file for configuring the DLKM
`system:`	The system file for configuring the DLKM

After going into the proper directory, run the following commands as root. /usr/bin is required in the path when running the config command. Even though the command may be specified using the full path, config is relying on /usr/sbin being in the path when it tries to execute the kmupdate command.

```
cd <DLKM directory>                  # See below for correct directory.
/usr/sbin/kminstall -d <DLKM Name> # See below for correct name.
/usr/sbin/kminstall -a <DLKM Name>
/usr/sbin/config -M <DLKM Name> -u
/usr/sbin/kmadmin -L <DLKM Name>    # Do this step only for gvid_info!
```

You can check the kernel message buffer to see if any of these commands failed. As root, run the command /etc/dmesg. As a reminder, the kmadmin command cannot be used to load the DRM or the hw_init module. If used, the command will fail. Xf86 loads these modules when it first starts.

loading a graphics DLKM on HP-UX 11.23 and later

All graphics DLKM s are loaded using the same commands. First change into the directory where the DLKM resides. This directory should contain one file:

<DLKM Name>: The actual DLKM, this is listed below.

After you are in the proper directory, run the following commands as root:

```
cd <DLKM directory>                  # See below for correct directory.
kcmodule -s <DLKM Name>=unused     # See below for correct name.
cp <DLKM Name> /usr/conf/mod
kcmodule -s <DLKM Name>=auto
kcmodule -s <DLKM Name>=loaded     # Do this step only for gvid_info!
```

You can check the kernel message buffer to see if any of the commands failed. As root, run the command /etc/dmesg.

The list that follows shows the name and directories of each of the graphics DLKMs:

```
gvid_info DLKM:
  DLKM Name: gvid_info
  /opt/graphics/common/kernel/gvid_info/hpux64
ATI Fire GL4 DRM DLKM:
  DLKM Name:  drmfgl
  /opt/graphics/common/kernel/fgldrm/hpux64
ATI FIre GL4 hw_init DLKM:
  DLKM Name:  gvid_him_fgl
 /opt/graphics/common/kernel/gvid_hw_init/firegl23/hpux64
ATI FireGL X1 and ATI FireGL Z1 DRM DLKM:
  DLKM Name:  drmfglrx
  /opt/graphics/common/kernel/fgldrmrx/hpux64
ATI FireGL X1 and ATI FireGL Z1 hw_init DLKM:
  DLKM Name:  gvid_him_rad
  /opt/graphics/common/kernel/gvid_hw_init/radeon/hpux64
```

4

configuring an X Server on HP-UX

This chapter documents information specific to the HP Xf86 X Server. The X Server is based on the XFree86 version 4.1.0 or later X Server. This section describes features unique to HP's implementation of the X Server, provides information on how to configure the X Server and includes a list of supported X configurations. For each supported graphics device, device-dependent configuration information is provided.

using SAM to configure X Server

Configuration of the X Server is supported through SAM via an icon titled “X Server Configuration.” This icon resides either at SAM’s top level or under the top-level “Display” icon.

The SAM graphical user interface for X Server configuration is provided to simplify modifying the X Server configuration file, XF86Config. The X Server uses the XF86Config file for its configurations. While it is still possible to modify this file manually (see the XF86Config file on page 4-2), using the SAM interface can greatly simplify the process.

The SAM component has the following actions. For more information on configuring the X Server and these actions, see the SAM online help.

global actions

These actions are typically active regardless of what has been selected. If any of these menu items are not visible it is because they are not supported under the current configuration.

- **Configure Print Server** lets you manage print servers. From this menu item you can create, stop or remove print servers
- **Modify Server Options** lets you specific X Server options. See the menu item for specific options.
- **Modify Multi-Screen Layout is grayed out and is not available.**
- **Single Logical Screen (SLS)** lets you create, modify, or undo your SLS configuration. SLS is a mechanism for treating multidisplay configurations as a single “logical” screen. This allows the moving or spanning of windows across multiple physical monitors. SLS configurations may include up to four screens. All component screens share the same input devices (for example, one keyboard and mouse for the SLS configuration).

screen actions

These actions are activated depending on which screens have been chosen.

- **Describe Screen** provides information about the device.
- **Identify Screen** flashes the monitor that is connected to the graphics device.
- **Modify Default Visual** lets you set the default visuals, depth and resolution on a graphics device. It lets you identify which of these should be the default settings.
- **Modify Screen Options** contains options that are specific to each graphics device. The options differ depending on the capabilities of each card.

other actions

Grayed out screen icons represent screens that have not been configured for use by the X Server. Use these commands to add and remove screens to and from the configuration file.

- **Add Screen to Configuration** lets you add grayed out screen icons to the configuration file. Select the grayed out icons and choose the `Add Screen to Configuration` menu item to add screens to the configuration file.
- **Remove Screen from Configuration** operates on configured screens. Each configured screen selected is removed from the configuration and becomes “Unused.” The X Server is not brought up on Unused screens. If this menu option is grayed out, it may be activated by selecting a configured screen. This menu item is not visible when there is only one screen present which must remain in the configuration.

using setmon to configure the monitor

`setmon` is a configuration tool used to change the settings for a monitor attached to a graphics device. This tool permits you to change the monitor's refresh rate, frame buffer resolution, and frame buffer memory configuration (for example, Stereo, Double Buffer), when the device supports multiple options. To change the monitor type, the `setmon` command can be executed directly or done through SAM.

The `setmon` executable is located at `/opt/graphics/common/bin/setmon`. Under SAM, this component is an icon called Monitor Configuration located under the top-level Display folder, next to the X Server Configuration icon.

NOTE: Changing the monitor type while the X Server is running requires stopping and restarting the X Server. To change the monitor settings, the X Server needs to be running on the device specified. For these graphics cards, it may not be possible to test some of the monitor settings before making the change permanent.

---

the XF86Config file

The XF86Config file is located in `/etc/X11/XF86Config`. It can be generated automatically or modified using SAM. A working configuration file is also delivered on the system. You must be root to create or edit this file. The XF86Config manual page provides additional information regarding the configuration file. For any changes made to the XF86Config file to take effect, it is necessary to restart the X Server.

XF86Config file format

Most of the content in this section has been copied from the XF86Config(5) manual page listed on “The XFree86 Project, Inc.” web site (www.xfree86.org). The manual pages are available from www.xfree86.org/4.1.0.

Configuration file keywords are case-insensitive, and underscore (_) characters are ignored. Most strings (including option names) are also case insensitive, and insensitive to white space and underscore “_” characters.

Each configuration file entry usually takes up a single line in the file. Each entry consists of a keyword, which is possibly followed by one or more arguments, with the number and types of the arguments depending on the keyword. The argument types are:

- **Integer** — an integer number in decimal, hex or octal
- **Real —** a floating point number
- **String** — a string enclosed in double quote marks (“)

NOTE: Hex integer values must be prefixed with “0x”, and octal values with “0”.

A special keyword called **Option** may be used to provide free-form data to various components of the server. The Option keyword takes either one or two string arguments. The first is the option name, and the optional second argument is the option value:

- **Integer** — an integer number in decimal, hex or octal
- **Real** — a floating point number
- **String** — a sequence of characters
- **Boolean** — a boolean value (see below)
- **Frequency** — a frequency value (see below)

NOTE: All Option values, not just strings, must be enclosed in quotes.

Boolean options may optionally have a value specified. When no value is specified, the option's value is TRUE. The following boolean option values are recognized as TRUE:

```
1, on, true, yes
```

and the following boolean option values are recognized as FALSE:

```
0, off, false, no
```

If an option name is prefixed with “No”, then the option value is negated.

Frequency option values consist of a real number that is optionally followed by one of the following frequency units:

```
Hz, k, kHz, M, MHz
```

When the unit name is omitted, the correct units are determined from the value and the expectations of the appropriate range of the value. It is recommended that the units always be specified when using frequency option values to avoid any errors in determining the value.

ServerLayout section

The **ServerLayout** section identifies which **Screen** sections are to be used in a multiheaded configuration, the relative layout of those screens, and which **InputDevice** sections are to be used. Each ServerLayout section has an **Identifier**, a list of **Screen** section identifiers, and a list of InputDevice section identifiers. Options may also be included in the ServerLayout section. A ServerLayout section may be made active by referencing (via its Identifier) on the command line that starts X. In the absence of this, the first one found in the file is chosen by default, as there may be multiple ServerLayout sections in the configuration file. The format of the ServerLayout section is as follows:

```
Section “ServerLayout”
   Identifier “ServerLayoutName”
   Screen [ScreenNumber] “ScreenID” [Position] [Xcoor] [Ycoor]
   . . .
   InputDevice  “InputDeviceID” “InputDeviceOption”
   . . .
  [Option …]
   . . .
EndSection
```

Keywords, options and values enclosed in [ ] are optional.

A number specifying the preferred screen number for that screen may optionally follow each Screen. When no screen number is specified, it is numbered according to the order in which it is listed. Next comes the **ScreenID**, a required field that must be enclosed in double quotes. The ScreenID must match an Identifier in a Screen section. The remaining information on the line is optional. Next comes the physical position of the screen, either in absolute terms or relative to another screen (or screens). Finally the XY coordinates of the screen may be specified.

The position keywords are:

Absolute
RightOf
LeftOf
Above
Below
Relative

The preferred method of specifying the layout is to explicitly specify the screen's location in absolute terms or relative to another screen.

The examples are based on the examples listed in the DESIGN document from XFree86.

In the absolute case, the upper left corner's coordinates are given after the **Absolute** keyword. If the coordinates are omitted, a value of (0,0) is assumed. An example of absolute positioning follows:

```
Section “ServerLayout”
      Identifier “MainLayout”
      Screen 0         “Screen 1" Absolute
      Screen 1         “Screen 2" Absolute 1024  0
      Screen           “Screen 3" Absolute 2048  0
      . . .
EndSection
```

When the **Relative** keyword is used, the coordinates of the new screen's origin relative to reference screen follow the reference screen name. The following example shows how to use some of the relative positioning options:

```
Section “ServerLayout”
      Identifier “MainLayout”
      Screen 0         “Screen 1" Absolute
      Screen 1         “Screen 2" Absolute 1024  0
      Screen           “Screen 3" Absolute 2048  0
      . . .
EndSection
```

Each **InputDevice** is followed by an InputDeviceID, a required field that must be enclosed in double quotes. The InputDeviceID must match an Identifier in an InputDevice section. Last, an option may be provided. The option can also be specified in the InputDevice section. Typical options specified here are: **CorePointer**, **CoreKeyboard**, and **SendCoreEvents**. The option must be enclosed in double quotes. See input devices on page 4-38 for more information regarding the options. Normally, at least two InputDevices are present: a keyboard and a mouse.

Options that apply to the X Server may also be specified in this section. The following table lists all options that may be set in the ServerLayout section. This information is from the XF86Config manual page.

Xserver Options for ServerLayout Section

Option	Value	Default	Description
DontZap	Boolean	Off	Disallows use of the **Ctrl+Shift+Break** sequence. That sequence is normally used to terminate the X Server. When this option is enabled, that key sequence has no special meaning and is passed to clients.
DontZoom	Boolean	Off	Disallows use of the **Ctrl+Alt+Keypad-Plus** and **Ctrl+Alt+Keypad-Minus** sequences. These sequences allow you to switch between video modes. When this option is enabled, those key sequences have no special meaning and are passed to clients.
AllowMouseOpenFail	Boolean	false	Allows the server to start up even if the mouse device can't be opened/initialized.
Pixmap	Bpp	32	Sets the pixmap format to use for depth 24. Allowed values for *bpp* are 24 and 32. Default: 32 unless driver constraints don't allow this (which is rare). **Note:** some clients don't behave well when this value is set to 24.
Verbose	Integer	-1	See logging and verbosity on page 4-19.
NoLogging	NA	NA	See logging and verbosity on page 4-19.

Xserver Options for ServerLayout Section (Continued)

Option	Value	Default	Description
LogVerbose	Integer	-1	See logging and verbosity on page 4-19.
CursorScaleFactor	Integer	1	See cursor scaling on page 4-19.
MaxCursorSize	Integer	64	See cursor scaling on page 4-19.
AccelerateIndirectRendering	Boolean	True	Specifies whether or not OpenGL is to do software rendering. A value of False forces software rendering. The default is for OpenGL to use accelerated rendering.

Files section

The **Files** section specifies paths to where fonts and modules are located and the location of the rgb database and the user specified logfile. The Files section format is:

```
"Files" Section
       [FontPath     "PathName"]
        .
        .
       [ModulePath   "PathName"]
        .
        .
       [RgbPath        "PathName"]
       [LogPath        "PathName"]
Endsection
```

Multiple **FontPaths** and **ModulePaths** may be specified either by multiple lines or by using a comma delimiter between paths on the same line.

FontPath elements may be either absolute directory paths, or a font server identifier. Font server identifiers have the form:

```
<trans>/<hostname>:<port-number>/<cataloguelist>
```

where <*trans*> is the transport type to use to connect to the font server (for example, **Unix** for UNIX® domain sockets or **tcp** for a TCP/IP connection), <*hostname*> is the hostname of the machine running the font server, and <*port-number*> is the port number that the font server is listening on (usually 7000).

The default **FontPath** is:

```
tcp/<systemname>:7000/,
/usr/lib/X11/fonts/hp_roman8/75dpi/,
/usr/lib/X11/fonts/iso_8859.1/100dpi/,
/usr/lib/X11/fonts/iso_8859.1/75dpi/,
/usr/lib/X11/fonts/hp_kana8/,
/usr/lib/X11/fonts/hp_japanese/100dpi/,
/usr/lib/X11/fonts/hp_japanese/75dpi/,
/usr/lib/X11/fonts/hp_korean/75dpi/,
/usr/lib/X11/fonts/hp_chinese_s/75dpi/,
/usr/lib/X11/fonts/hp_chinese_t/75dpi/,
/usr/lib/X11/fonts/iso_8859.2/75dpi/,
/usr/lib/X11/fonts/iso_8859.5/75dpi/,
/usr/lib/X11/fonts/iso_8859.6/75dpi/,
/usr/lib/X11/fonts/iso_8859.7/75dpi/
/usr/lib/X11/fonts/iso_8859.8/75dpi/,
/usr/lib/X11/fonts/iso_8859.9/75dpi/,
/usr/lib/X11/fonts/misc/
```

The X Server uses ModulePaths as locations to look for loadable modules. The default ModulePath is:

```
/usr/lib/hpux32/X11/Xserver/modules/xf86,
/opt/graphics/common/lib/hpux32
```

When FontPath or ModulePath are specified in the configuration file, they override the default values.

RgbPath can be used to specify the RGB database path. Normally it is never changed. If it is not specified the built-in path `/etc/X11/rgb` is used.

In addition, the **LogPath** can be specified, if server logging information is to be sent somewhere other than the default log file. The default log file is located at `/var/X11/Xserver/logs/Xf86.n.log`, where *n* is the display number.

All names must be enclosed within double quotes. There may be only one Files section in the configuration file. This section does not recognize Option as a keyword.

Module section

The **Module** section specifies which X Server modules should be loaded. The types of modules normally loaded in this section are X Server extension modules, and font rasterizer modules. Most other module types are loaded automatically when they are needed via other mechanisms. There may only be one Module section in the configuration file. The format of the Module section is as follows:

```
Section “Module”
        Load “ModuleName”
         . . .
        [SubSection “ModuleName”
            Option . . .
             . . .
        EndSubSection]
         . . .
EndSection
```

Load instructs the server to load the module called **ModuleName**. The module name given should be the module's extension name, not the module file name. The extension name is case sensitive, and does not include the “lib” prefix, or the library suffix (for example, “so.1”).

Example: the Double Buffered Extension (DBE) can be loaded with the following entry:

```
Load “dbe”
```

Beginning with the December 2002 patch, the “Load” directive is no longer needed to load extensions because the brokering mechanism ensures that the correct extensions are loaded automatically.

SubSection also instructs the server to load the module called ModuleName. The module name given should be the module's extension name, not the module file name. The extension name is case sensitive, and does not include the “lib” prefix, or the library suffix (for example, “so.1”). The difference is that the listed Options are passed to the module when it is loaded.

Modules are searched for in each directory specified in the ModulePath search path and in the drivers, input, extensions, fonts, and HP-UX subdirectories of each directory in the ModulePath. If ModulePath is not specified in the Files section, the default ModulePath is searched.

Noload instructs the server to not load the module called ModuleName.

InputDevice section

An **InputDevice** section is considered active if there is a reference to it in the active ServerLayout section. There may be multiple InputDevice sections. There are normally at least two: one for the core (primary) keyboard, and one for the core pointer. InputDevice sections have the following format:

```
Section "InputDevice"
        Identifier      "InputDeviceID"
        Driver          "DriverName"
        [Option …]
        . . .
EndSection
```

The **Identifier** entry specifies the unique name for this input device and must match an InputDeviceID in the active ServerLayout section in order to be active.

The **Driver** entry specifies the name of the driver to use for this input device.

InputDevice sections recognize some driver-independent Options, which are described here. See the individual input driver manual pages for a description of the device-specific options that can be entered here.

Options for InputDevice Section

Option	Value	Description
CorePointer	NA	When this is set, the input device is installed as the core (primary) pointer device. There must be no more than one core pointer. If this option is not set here, or in the **ServerLayout** section, or from the -pointer command line option, then the first input device that is capable of being used as a core pointer is selected as the core pointer. Source: XF86Config manual page.
CoreKeyboard	NA	When this is set, the input device is to be installed as the core (primary) keyboard device. There must be no more than one core keyboard. If this option is not set here, or in the **ServerLayout** section, then the first input device that is capable of being used as a core keyboard is selected as the core keyboard. Source: XF86Config manual page.
AlwaysCoreSendCoreEvents	boolean	Both of these options are equivalent, and when enabled cause the input device to always report core events. This can be used, for example, to allow additional pointer devices to generate core pointer events (such as moving the cursor, etc). Source: XF86Config manual page.
HistorySize	integer	Sets the motion history size. Default: 0. Source: XF86Config manual page.

The following two examples show an **InputDevice** section for a keyboard and mouse:

```
Section “InputDevice”
        Identifier    “Keyboard0”
        Driver        “keyboard”
EndSection
Section “InputDevice”
        Identifier     “Mouse0”
        Driver         “mouse”
        Option         “Protocol”   “PS/2”
         Option         “Device”   “/dev/hid/mouse_000”
EndSection
```

The above **InputDevice** section can be modified to run the X Server without a mouse. To do this, simply change the “Device” option specification in the mouse section to:

```
        Option         “Device”    “NULL”
```

The same specification can be added to the keyboard section to instruct the X Server to ignore a keyboard attached to the system.

Screen section

The configuration file may have multiple **Screen** sections. There must be at least one, for the “screen” being used. A “screen” binds a graphics device (Device section) and a monitor (Monitor section) together. A Screen section is considered “active” if it is referenced by an active ServerLayout section. If neither of these is present, the first Screen section found in the configuration file is considered the active one. Screen sections have the following format:

```
Section “Screen”
       Identifier “ScreenID”
       Device     “DeviceID”
       Monitor    “MonitorID”
       DefaultDepth   <Depth>
       Option ...
       .
       .
       SubSection “Display”
              .
              .
       EndSubSection
       .
EndSection
```

The **Identifier** entry specifies the unique name for this screen. The Identifier generally must match a ScreenID listed in the active ServerLayout section. The Screen section provides information specific to the whole screen, including screen-specific Options. In multiscreen configurations, there are multiple active Screen sections, one for each head.

The Device keyword specifies which Device section is used for this screen. This is what binds a specific graphics card to a screen. The DeviceID must match the Identifier of a Device section in the configuration file.

The Monitor keyword specifies which Monitor section is used for this screen. This is what binds a specific monitor to the screen. The MonitorID must match the Identifier of a **Monitor** section in the configuration file.

The DefaultDepth keyword specifies which color depth the server should use by default. The `-depth` command line option can be used to override this. If neither is specified, the default depth is driver specific, but in most cases is 8.

Various Option flags may be specified in the Screen section. Some are driver specific and are described in the driver documentation. Driver-independent options are described here. This information is from the XF86Config manual page.

Options for Screen Section

Entry	Entry Position	Description
Accel	NA	Enables XAA (X Acceleration Architecture), a mechanism that makes video cards' 2D hardware acceleration available to the Xserver. This option is on by default. There are many options to disable specific accelerated operations, listed below. Note that disabling an operation has no effect if the operation is not accelerated (whether due to lack of support in the hardware or in the driver).
XaaNoCPUToScreenColor ExpandFill	NA	Disables accelerated rectangular expansion bits from source patterns stored in system memory (using a memory-mapped aperture).
XaaNoColor8x8PatternFillRect	NA	Disables accelerated fills of a rectangular region with a full-color pattern.
XaaNoColor8x8PatternFillTrap	NA	Disables accelerated fills of a trapezoidal region with a full-color pattern.
XaaNoDashedBresenhamLine	NA	Disables accelerated dashed Bresenham line draws.
XaaNoDashedTwoPointLine	NA	Disables accelerated dashed lines drawn between two arbitrary points.
XaaNoImageWriteRect	NA	Disables accelerated transfers of full-color rectangular patterns from system memory to video memory (using a memory-mapped aperture).
XaaNoMono8x8PatternFillRect	NA	Disables accelerated fills of a rectangular region with a monochrome pattern.

Options for Screen Section (Continued)

Entry	Entry Position	Description
XaaNoMono8x8PatternFillTrap	NA	Disables accelerated fills of a trapezoidal region with a monochrome pattern.
XaaNoOffscreenPixmaps	NA	Disables accelerated draws into pixmaps stored in offscreen video memory.
XaaNoPixmapCache	NA	Disables caching of patterns in offscreen video memory. Source: XF86Config manual page.
XaaNoScanlineCPUToScreen ColorExpandFill	NA	Disables accelerated rectangular expansion blits from source patterns stored in system memory (one scan line at a time).
XaaNoScanlineImageWriteRect	NA	Disables accelerated transfers of full-color rectangular patterns from system memory to video memory (one scan line at a time).
XaaNoScreenToScreen ColorExpandFill	NA	Disables accelerated rectangular expansion blits from source patterns stored in offscreen video memory.
XaaNoScreenToScreenCopy	NA	Disables accelerated copies of rectangular regions from one part of video memory to another part of video memory.
XaaNoSolidBresenhamLine	NA	Disables accelerated solid Bresenham lines drawn.
XaaNoSolidFillRect	NA	Disables accelerated solid-color fills of rectangles.
XaaNoSolidFillTrap	NA	Disables accelerated solid-color fills of Bresenham trapezoids.
XaaNoSolidHorVertLine	NA	Disables accelerated solid horizontal and vertical lines.
XaaNoSolidTwoPointLine	NA	Disables accelerated solid lines drawn between two arbitrary points.
SuppressVisuals	string	See Glx visual suppression on page 4-20.
SuppressGlxVisuals	string	See Glx visual suppression on page 4-20.

Each Screen section must contain one or more **Display** subsections. Those subsections provide depth configuration information, and the one chosen depends on the depth that is being used for the screen. The Display subsection format is described in the section below.

Display subsection

The **Screen** sections include one or more **Display** subsections. One Display subsection may be provided for each depth that the server supports. The size of the virtual screen may also be specified. The virtual screen allows you to have a “root window” larger than what can be displayed on the monitor. (e.g. the monitor may be a 800x600 display, but have a 1280x1024 virtual size). The **Virtual** keyword specifies this size. Note that many of the new accelerated graphics drivers use nondisplayed memory for caching. It is not desirable to use all available memory for the virtual display, as this leaves none for caching, and this can decrease server performance. Display subsections have the following format:

```
SubSection “Display”
     Depth     depth
     Visual    visual
     Modes    “ModeName”
     ViewPort  x0  y0
     Option...
     ...
EndSubSection
```

The **Depth** entry specifies what color depth the Display subsection is to be used for. Only Depths of 8 and 24 are supported.

The **Modes** entry specifies the list of video modes to use. Each **ModeName** specified must be in double quotes. They must correspond to those specified or referenced in the appropriate **Monitor** section. The server deletes modes from this list which don't satisfy various requirements. The first valid mode in this list is the default display mode for startup. The list of valid modes is converted internally into a circular list. It is possible to switch to the next mode with **Ctrl+Alt+Keypad-Plus** and to the previous mode with **Ctrl+Alt+Keypad-Minus**. When this entry is omitted, the largest valid mode referenced by the appropriate Monitor section is used.

The **Visual** entry is optional and sets the default root visual type.

The only visual type available for depth 8 is: PseudoColor

The only visual type available for depth 24 is: TrueColor

The **ViewPort** entry is optional and sets the upper left corner of the initial display. This is only relevant when the virtual screen resolution is different from the resolution of the initial video mode. If this entry is not given, then the initial display is centered in the virtual display area.

Option flags may be specified in the Display subsections. These may include driver-specific options or driver-independent options. The former are described in the driver-specific documentation. Some of the latter are described above in the Screen section, and they may also be included here. However, options set in the Display subsection may be “overridden” in the Screen section.

Monitor section

The configuration file may have multiple **Monitor** sections. The Monitor section provides information about the specifications of the monitor, monitor-specific Options, and information about the video modes to use with the monitor. There must be at least one Monitor section, for the monitor being used. A Monitor section is considered “active” if it is referenced by an active Screen section. Monitor sections have the following format:

```
Section “Monitor”
    Identifier   “MonitorID”
    VendorName   “Vname”
    ModelName    “Mname”
    HorizSync     horizsync-range
    VertRefresh   vertrefresh-range
    DisplaySize   width  height
    Gamma    [gamma-value|{red-gamma green-gamma blue-gamma}]
EndSection
```

The **Identifier** entry specifies the unique name for this monitor.

The **VendorName** is an optional entry and specifies the monitor's manufacturer.

The **ModelName** is an optional entry that specifies the monitor model.

HorizSync gives the range(s) of horizontal sync frequencies supported by the monitor. *horizsync-range* may be a comma separated list of either discrete values or ranges of values. A range of values is two values separated by a dash. By default the values are in units of kHz. They may be specified in MHz or Hz if **MHz** or **Hz** is added to the end of the line. The data given here is used by the X Server to determine if video modes are within the specifications of the monitor. This information should be available in the monitor's handbook. If this entry is omitted, a default range of 28-33 kHz is used.

VertRefresh gives the range(s) of vertical refresh frequencies supported by the monitor. *vertrefresh-range* may be a comma separated list of either discrete values or ranges of values. A range of values is two values separated by a dash. By default the values are in units of Hz. They may be specified in MHz or kHz if MHz or kHz is added to the end of the line. The data given here is used by the X Server to determine if video modes are within the specifications of the monitor. This information should be available in the monitor's handbook. If this entry is omitted, a default range of 43-72Hz is used.

DisplaySize is an optional entry giving the width and height, in millimeters, of the picture area of the monitor. If given these values are used to calculate the horizontal and vertical pitch (DPI) of the screen.

Gamma is an optional entry that can be used to specify the gamma correction for the monitor. It may be specified as either a single value or as three separate rgb values. The values should be in the range 0.1 to 10.0, and the default is 1.0. Not all drivers are capable of using this information.

Device section

The configuration file may have multiple **Device** sections. There must be at least one, for the video card being used. Device sections have the following format:

```
Section “Device”
     Identifier     “DeviceID”
     Driver         “driver”
     VendorName     “Vname”
     [BusID         “busid”]
     Option …
     . . .
EndSection
```

The **Identifier** entry specifies the unique name for this graphics device. It must match a DriverID in the active Screen section.

The **Driver** entry specifies the name of the driver to use for this graphics device. When using the loadable server, the driver module “driver” is loaded for each active Device section.

Supported drivers are:

```
firegl23, radeon
```

The keyword **BusID** specifies the bus location of the graphics card. For PCI/AGP cards, the *busid* string has the form `PCI:bus:device:function`. This field is optional in single-head configurations when using the primary graphics card. In multi-head configurations, or when using a secondary graphics card in a single-head configuration, this entry is mandatory. Its main purpose is to make an unambiguous connection between the **Device** section and the hardware it is representing.

To find the bus ID of the graphics device you can use the command

```
ioscan -f | grep gvid
```

Beginning with the December, 2002 X Server patch, the brokering mechanism is supported and the Driver and BusID keywords are no longer supported in the “Device” section. They have been replaced by the Devicefile keyword. The Devicefile keyword identifies which character device provides the interface with the actual hardware. The device should be a `/dev/gvid device` (for example, `/dev/gvid0, /dev/gvid1`)

Options are device dependant and should be searched for in device specific sections of this document.

extensions

double buffer extension (DBE)

DBE is an extension to the X Server that provides a double-buffering API. For more information about DBE and the API, consult the DBE manual pages:

- DBE
- XdbeQueryExtension
- XdbeGetVisualInfo
- XdbeFreeVisualInfo
- XdbeScreenVisualInfo
- XdbeAllocateBackBufferName
- XdbeDeallocateBackBufferName
- XdbeSwapBuffers
- XdbeBeginIdiom
- XdbeEndIdiom
- XdbeGetBackBufferAttributes

determining swap performance

The DBE API does not allow users to determine if double buffering in a visual is through software or hardware. However, the API does provide a way to determine relative swapping performance on a per visual basis. The XdbeScreenVisualInfo() function returns information about the swapping performance levels for the double-buffering visuals on a display. A visual with a higher performance level is likely to have better double-buffer graphics performance than a visual with a lower performance level. Nothing can be deduced from any of the following: the magnitude of the difference of two performance levels, a performance level in isolation, or comparing performance levels from different servers.

display power management signaling (DPMS)

Monitors constitute a large percentage of the power used by a workstation even when not actively in use (i.e. during screen blanking). In order to reduce the power consumption, the Video Electronic Standards Association (VESA) has defined a Display Power Management Signaling (DPMS) standard which can be used to greatly reduce the amount of power being used by a monitor during screen blanking.

The following table is a description of the states that are defined by VESA. The Power Savings column indicates (roughly) the level of power savings achieved in the given state. The Recovery Time is the amount of time that the screen takes to return to a usable state when the screen saver is turned off (for example, by pressing a key or by moving the mouse).

States defined by VESA

Level	State	DPMS Compliance Requirements	Power Savings	Recovery Time
0	Screen Saver	Not Applicable	None	Very Short (<1 sec)
1	Standby	Optional	Minimal	Short
2	Suspend	Mandatory	Substantial	Longer
3	Off	Mandatory	Maximum	System Dependent

The actual amount of power saved and the recovery time for each of the states is monitor dependent and may vary widely. The customer can compensate for this by choosing an appropriate level for the monitor that is currently in use.

By default, the DPMS level used is the Screen Saver (i.e. no power savings). If you wish to use power saving during screen blanking, set the following XF86Config file entries before starting the server: `blank time`, `standby time`, `suspend time`, and `off time`.

The DPMS Extension lets individual users customize their personal DPMS settings to meet their work styles and any restrictions imposed by their employers. For example, an employer may decide that all monitors must save power after 30 minutes of idle time. The individual user may decide that 30 minutes is too long, and adjust the time downward to meet their work preference.

More information (including sample code) on the DPMS Extension entry points can be found online, via the manual pages. The extension entry points are:

```
DPMS
DPMSQueryExtension
DPMSGetVersion
DPMSCapable
DPMSSetTimeouts
DPMSGetTimeouts
DPMSEnable
DPMSDisable
DPMSForceLevel
DPMSInfo
```

XFree86 provides four options that may be set in the ServerLayout section that may be used to support this functionality. The options are: blank time, standby time, suspend time, and off time. The following example sets these to 10, 20, 30, and 60 minutes respectively.

```
Section “ServerLayout”
    . . .
     Option     “BlankTime”          “10”
     Option     “StandbyTime”        “20”
     Option     “SuspendTime”        “30”
     Option     “OffTime”            “60”
    . . .
EndSection
```

Options for ServerLayout Section

Option	Value	Default	Description
BlankTime	time	10	Sets the inactivity timeout for the blanking phase of the screensaver. *Time* is in minutes. This is equivalent to the Xserver's `-s' flag, and the value can be changed at run time with `xset(1)`.
StandbyTime	time	20	Sets the inactivity timeout for the "standby" phase of DPMS mode. *Time* is in minutes, and the value can be changed at run time with `xset(1)`.
SuspendTime	time	30	Sets the inactivity timeout for the "suspend" phase of DPMS mode. *Time* is in minutes, and the value can be changed at run time with `xset(1)`.
OffTime	time	40	Sets the inactivity timeout for the "off" phase of DPMS mode. *Time* is in minutes, and the value can be changed at run time with `xset(1)`.

dynamic library loading

The path for each dynamically loaded module must be specified in the **ModulePath** in order for them to load. See the Module section on page 4-8 for more details regarding the ModulePath.

Dynamically loaded modules are recorded by the X Server in the `/var/X11/Xserver/logs` directory. The log file reflects the display identifier for a given run. Only the last invocation against a given display identifier is retained. The log file contains the parsed contents of the XF86Config file and the full path name for all dynamically loaded modules for the given X Server invocation. Deferred loaded modules are recorded as they are referenced.

NOTE: Altering or removing files under `/usr/lib/X11/Xserver` prevents the Technical Print Server from running. Altering or removing files under `/usr/lib/hpux32/X11/Xserver/modules/xf86` prevents the X Server from running.

features

cursor scaling

At times the standard X11 cursors are difficult to see on the screen. The effect is compounded on large displays. Two options are available in the X Server that instruct the X Server to scale all X11 cursors (both user-defined and built-in cursors) by a user-defined value.

Cursor Scaling is indicated with the following syntax in the XF86Config file:

```
Section "ServerLayout"
     . . .
     Option     "CursorScaleFactor"    "n"
     Option     "MaxCursorSize"       "Size"
     . . .
EndSection
```

Where n = 1, 2, 3, …

Where Size = 2, 4, 8, 16, 32, 64, 128

For example, n=2 instructs the X Server to scale all cursors by "2x" so that a 16x16 cursor becomes a 32x32 cursor and a 9x9 cursor becomes an 18x18 cursor, etc.

If the scaled width or height of any cursor is greater than Size, the scale factor is reduced so that the net size of the cursor fits into a Size x Size rectangle.

The default value for "n" is 1, or no scaling. The default value for "Size" is 64, or 64x64 maximum size.

logging and verbosity

Four options are available to control logging in the XF86Config file: **Verbose**, **LogVerbose**, **NoLogging**, and **LogFile**. The first three are located in the ServerLayout section and the last one is set in the Files section.

- **Verbose** controls what is written to `stderr`. A negative option value effectively shuts off anything written to `stderr`, except errors and fatal errors. As values increase from 0 to 5, more information is written to `stderr`. Increasing the value beyond 5 has no effect. The default value is 1.
- **LogVerbose** controls how much is written to the log file. A negative option value effectively shuts off anything being written to the log file, except errors and fatal errors. As values increase from 0 to 5, more information is written to the log file. Increasing the value beyond 5 has no effect. The default value is 3.
- **NoLogging** closes the default log file and uses `/dev/null` as the log file. Therefore, anything being written to the log file is lost. The default value allows logging to the log file, either the default log file located in `/var/X11/Xserver/logs` or the user specified LogPath.
- **LogFile** closes the built-in default log file and opens the file specified by this option as the default log file, unless the NoLogging option is selected. The NoLogging option overrides this option.

The following is an example of setting the various logging options. The Verbose and LogVerbose options are set so that only error and fatal error messages are sent to `stderr` and the log file. The NoLogging option uses `/dev/null` to be the log file. It overrides the LogFile specification in the Files section. Commenting out NoLogging opens `/tmp/Xlog.log` and sends all messages directed to this log file.

```
Section “ServerLayout”
     . . .
     Option     “Verbose”      “-1”
     Option     “LogVerbose”   “-1”
     Option     “NoLogging”
     . . .
EndSection
Section “Files”
     . . .
     LogFile     “/tmp/Xlog.log”
     . . .
EndSection
```

Glx visual suppression

This option “hides” visuals. It reduces the number of visuals made available to clients. The example that follows demonstrates how to suppress all visuals except for the most capable of each class of visuals.

```
Section “Section”
. . .
Option  “SuppressGlxVisuals”  “HideDuplicateGlxVisuals”
. . .
EndSection
```

The user can also selectively “hide” classes of visuals. For example, to suppress any visual that has either Alpha planes or Stencil planes, do:

```
Option  “SuppressGlxVisuals”  “NoAlpha & NoStencil”
```

The options can be white space, ampersand (&), or comma (,) delimited, and must be enclosed with a single pair of double quotes.

The following is a complete list of the classes of visuals that can be suppressed.

- IsRgba – RGB (True Color) visuals
- IsCi – Color Indexed (Pseudo Color) visuals
- Alpha – visuals that have Alpha planes.
- NoAlpha – visuals that don't have Alpha planes.
- Back – double buffered visuals
- NoBack – visuals that cannot be double buffered
- Accum – visuals that have an accumulation buffer
- NoAccum – visuals that don't have an accumulation buffer
- Depth – visuals that have a Z (depth) buffer
- NoDepth – visuals that don't have a Z (depth) buffer
- Stencil –visuals that have Stencil planes
- NoStencil – visuals that don't have Stencil planes
- Stereo – visuals that have Stereo buffers
- NoStereo – visuals that don't have Stereo buffers

If an opposing pair of options is selected (for example, Stereo and NoStereo) the suppress options is ignored because selecting an opposing pair would suppress all the visuals.

The **SuppressVisuals** option can be used to hide whole classes of visuals. These can be selected via the **SuppressVisuals** option in the Screen section of the XF86Config file. The visuals that can be suppressed are: PseudoColor and TrueColor. If all visuals are suppressed and no default visual has been selected, the X Server exits with a fatal error. Also if a default visual is defined, it overrides the suppression request. The example below suppresses the PseudoColor visual. The options can be delimited by white space, bar (|), or comma (,).

```
Section "Screen"
   Option  "SuppressVisuals" "PseudoColor"
EndSection
```

technical print service (TPS)

The Technical Print Service, `tps(5)`, is a network transparent printing system that allows X applications to render to non screen devices in the same manner they render to displays. It may also be referenced as the X Print Service. Please refer to the `tps(5)` manual page for details on configuring and using TPS.

virtual frame buffer (Xvfb)

`Xvfb(1)` is an X Server that does not require display hardware or input devices. It emulates a video frame buffer by using the system's virtual memory.

Xvfb may be used for: rendering with non-standard depths and screen configurations, software rendering, providing a way to run applications that don't need an X Server but for some reason insist on having one, etc.

Generally the user application must use functions such as `XGetImage(3)` in order to see what was rendered.

Most of the content in this section is based on the `Xvfb(1)` manual page provided by XFree86.

configuring the virtual frame buffer

No configuration file is required to use the virtual frame buffer.

using the virtual frame buffer

In addition to the normal server options described in "Starting the X Server From the Command Line", Xvfb accepts the following command line switches.

Command Line Switches

Switch	Value	Description
-screen	ScrnNum WxHxD	Creates a screen and sets its width, height, and depth to W, H, and D respectively. By default ScrnNum is 0 and the default dimensions are: 1280x1024x8.
-pixdepths	Depth1 Depth2 . . .	Specifies a list of pixmap depths that the server should support in addition to the depths implied by the supported screens. The option is a space delimited list of integers ranging in value from 1 to 32.
-fbdir	FrameBufferDirectory	Specifies the directory in which the memory mapped file containing the frame buffer memory should be created. The created file is located at `/FramebufferDirectory/Xvfb_screen<ScrnNum>.` The file is in xwd format. Capturing a full-screen snapshot can be done with a file copy command. The resulting image contains the cursor image. If this option and the `–shmem` option are not specified, the frame buffer memory is allocated with `malloc()`.
-shmem	NA	Specifies that the frame buffer should be put in shared memory. The shared memory ID for each screen is printed by the server. The shared memory image is in xwd format. If this option and the –fbdir option are not specified, the frame buffer memory is allocated with `malloc()`.
-linebias	Bitmask	This option specifies how to adjust the pixelization of thin lines. The Bitmask value is interpreted to be in octants. The value represents the direction in which it is preferred to take an axial step when the Bresenham error term is exactly zero.

Command Line Switches (Continued)

Switch	Value	Description
-blackpixel	Value	Specifies the black value the server should use.
-whitepixel	Value	Specifies the white value the server should use.

Virtual Frame Buffer Examples

The following example starts a Virtual Frame buffer server that listens for connections as server display 100, at screen 0, a depth of 32, and screen dimensions of 1600x1200. Xvfb should not be started with a server number used by video or print servers, unless it is known the system does not run video or print servers.

```
Xvfb :100 -screen 0 1600x1200x32
```

The following example starts a server that listens for connections as server display 100, has the default screen configuration (one screen, 1280x1024x8), supports pixmap depths of 3 and 27, and uses memory mapped files in `/var/tmp` for the frame buffer.

```
Xvfb :100 -pixdepths 3 27 -fbdir /var/tmp
```

security

This entire section was copied from the XFree86 Xserver(1) manual page.

A security file must be placed somewhere on the system and loaded from the command line using the -sp option. The remainder of this section details the format and use of the security file.

The syntax of the security policy file is as follows. Notation: "*" means zero or more occurrences of the preceding element, and "+" means one or more occurrences. To interpret <foo/bar>, ignore the text after the /; it is used to distinguish between instances of <foo> in the next section.

```
<policy file>          ::= <version line> <other line>*
<version line>         ::= <string/v> '\n'
<other line >          ::= <comment> |
                           <access rule>|
                           <site policy>|
                           <blank line>
<comment>              ::= # <not newline>* '\n'
<blank line>           ::= <space> '\n'
<site policy>          ::= sitepolicy <string/sp> '\n'
<access rule>           ::= property <property/ar> <window>
                           <perms> '\n'
<property>             ::= <string>
<window>               ::= any | root | <required property>
<required property>    ::= <property/rp>|<property with value>
<property with value>  ::= <property/rpv> = <string/rv>
<perms>                ::= [<operation>|<action>|<space> ]*
<operation>            ::= r | w | d
<action>               ::= a | i | e
<string>               ::= <dbl quoted string> |
                             <single quoted string> |
                           <unquoted string>
<dbl quoted string>    ::= <space> " <not dqoute>* " <space>
<single quoted string>::= <space> ' <not squote>* ' <space>
<unquoted string>      ::= <space> <not space>+ <space>
<space>                 ::= [ ' ' | '\t' ]*

Character sets:

<not newline>::= any character except '\n'
<not dqoute> ::= any character except "
<not squote> ::= any character except '
<not space>   ::= any character except those in <space>
```

The semantics associated with the above syntax are as follows.

Security Section Syntax

Syntax	Description
<version line>	The first line in the file, specifies the file format version. If the server does not recognize the version <string/v>, it ignores the rest of the file. The version string for the file format described here is "version-1". Once past the *<version line>*, lines that do not match the above syntax are ignored.
<comment>	These lines are ignored.

Security Section Syntax (Continued)

Syntax	Description
<sitepolicy>	These lines are currently ignored. They are intended to specify the site policies used by the XC-QUERY-SECURITY-1 authorization method.
<access rule>	These lines specify how the server should react to untrusted client requests that affect the X Window property named *<property/ar>*. The rest of this section describes the interpretation of an *<access rule>*. For an *<access rule>* to apply to a given instance of *<property/ar>*, *<property/ar>* must be on a window that is in the set of windows specified by <window>. If <window> is any, the rule applies to *<property/ar>* on any window. If *<window>* is root, the rule applies to *<property/ar>* only on root windows. If *<window>* is *<required property>*, the following apply. If *<required property>* is a *<property/rp>*, the rule applies when the window also has that *<property/rp>*, regardless of its value. If <required property> is a *<property with value>*, *<property/rpv>* must also have the value specified by *<string/rv>*. In this case, the property must have type STRING and format 8, and should contain one or more null terminated strings. If any of the strings match *<string/rv>*, the rule applies. The definition of string matching is simple case-sensitive string comparison with one elaboration: the occurrence of the character '*' in *<string/rv>* is a wildcard meaning "any string." A *<string/rv>* can contain multiple wildcards anywhere in the string. For example, "x*" matches strings that begin with x, "*x" matches strings that end with x, "*x*" matches strings containing x, and "x*y*" matches strings that start with x and subsequently contain y. There may be multiple *<access rule>* lines for a given *<property/ar>*. The rules are tested in the order that they appear in the file. The first rule that applies is used.
<perms>	Specify operations that untrusted clients may attempt, and the actions that the server should take in response to those operations.

Security Section Syntax (Continued)

Syntax	Description
<operation>	Can be r (read), w (write), or d (delete). The following table shows how X Protocol property requests map to these operations in The Open Group server implementation. GetProperty r, or r and d if delete = True ChangeProperties w RotateProperties r and w DeleteProperty d ListProperties none, untrusted clients can always list all properties
<action>	Can be a (allow), i (ignore), or e (error). Allow means execute the request as if it had been issued by a trusted client. Ignore means treat the request as a no-op. In the case of **GetProperty**, ignore means return an empty property value if the property exists, regardless of its actual value. Error means do not execute the request and return a BadAtom error with the atom set to the property name. Error is the default action for all properties, including those not listed in the security policy file. An *<action>* applies to all <operation>s that follow it, until the next *<action>* is encountered. Thus, irwad means ignore read and write, allow delete. **GetProperty** and **RotateProperties** may do multiple operations (r and d, or r and w). If different actions apply to the operations, the most severe action is applied to the whole request; there is no partial request execution. The severity ordering is: allow < *ignore* < *error.* Thus, if the *<perms>* for a property are ired (ignore read, error delete), and an untrusted client attempts GetProperty on that property with delete = True, an error is returned, but the property value is not. Similarly, if any of the properties in a **RotateProperties** do not allow both read and write, an error is returned without changing any property values.

security example

```
version-1
# Allow reading of application resources, but not writing.

      property RESOURCE_MANAGER       root  ar iw
      property SCREEN_RESOURCES       root  ar iw

# Ignore attempts to use cut buffers. Giving errors causes
# apps to crash, and allowing access may give away too much
# information.

      property CUT_BUFFER0             root      irw
      property CUT_BUFFER1             root      irw
      property CUT_BUFFER2             root      irw
      property CUT_BUFFER3             root      irw
      property CUT_BUFFER4             root      irw
      property CUT_BUFFER5             root      irw
      property CUT_BUFFER6             root      irw
      property CUT_BUFFER7             root      irw

# Use these if you are using Motif.

      property _MOTIF_DEFAULT_BINDINGS   root     ar iw
      property _MOTIF_DRAG_WINDOW        root     ar iw
      property _MOTIF_DRAG_TARGETS       any      ar iw
      property _MOTIF_DRAG_ATOMS         any      ar iw
      property _MOTIF_DRAG_ATOM_PAIRS    any      ar iw

# The next two rules let xwininfo -tree work when untrusted.

      property WM_NAME                   any      ar

# Allow read of WM_CLASS, but only for windows with WM_NAME.
# This might be more restrictive than necessary, but
# demonstrates the <required property> facility, and is also
# an attempt to say “top level windows only.”
      property WM_CLASS                    WM_NAME   ar
# These next three let xlsclients work untrusted.Think
# carefully before including these; giving away the client
# machine name and command may be exposing too much.

      property WM_STATEWM_NAMEar
      property WM_CLIENT_MACHINE         WM_NAME   ar
      property WM_COMMAND                WM_NAME   ar

# To let untrusted clients use the standard colormaps created
# by xstdcmap, include these lines.

      property RGB_DEFAULT_MAP      root      ar
      property RGB_BEST_MAP         root      ar
      property RGB_RED_MAP          root      ar
      property RGB_GREEN_MAP        root      ar
```

```
        property RGB_BLUE_MAP         root      ar
        property RGB_GRAY_MAP         root      ar

    # To let untrusted clients use the color management database
    # created by xcmsdb, include these lines.

        property XDCCC_LINEAR_RGB_CORRECTION   root    ar
        property XDCCC_LINEAR_RGB_MATRICES     root    ar
        property XDCCC_GRAY_SCREENWHITEPOINT   root    ar
        property XDCCC_GRAY_CORRECTION         root    ar

    # To let untrusted clients use the overlay visuals that many
    # vendors support, include this line.

        property SERVER_OVERLAY_VISUALSroot    ar

    # Oddball property names and explicit specification of error
    # conditions.

        property “property with spaces” 'property with “'aw er ed

    # Allow deletion of Woo-Hoo if window also has property OhBoy
    # with value ending in “son”.  Reads and writes will cause an
    # error.

        property Woo-Hoo       OhBoy = “*son”ad
```

connecting to the network

The X Server supports client connections via a platform-dependent subset of the following transport types: TCP/IP and UNIX Domain sockets.

granting access

This section comes directly from the `XFree86 Xserver(1)` manual page.

The X Server implements a platform-dependent subset of the following authorization protocols: MIT-MAGIC-COOKIE-1, XDM-AUTHORIZATION-1, SUN-DES-1, and MIT-KERBEROS-5. See the `Xsecurity(1)` manual page for information on the operation of these protocols.

Authorization data required by the above protocols is passed to the server in a private file named with the `–auth` command line option. Each time the server is about to accept the first connection after a reset (or when the server is starting), it reads this file. If this file contains any authorization records, the local host is not automatically allowed access to the server, and only clients which send one of the authorization records contained in the file in the connection setup information is allowed access. See the Xau manual page for a description of the binary format of this file. See `xauth(1)` for maintenance of this file, and distribution of its contents to remote hosts.

The X Server also uses a host-based access control list for deciding whether or not to accept connections from clients on a particular machine. If no other authorization mechanism is being used, this list initially consists of the host on which the server is running as well as any machines listed in the file `/etc/Xn.hosts`, where *n* is the display number of the server. Each line of the file should contain either an Internet hostname (e.g. `expo.lcs.mit.edu`) or a DECnet hostname in double colon format (e.g. `hydra::`). There should be no leading or trailing spaces on any lines.

For example:

```
joesworkstation
corporate.company.com
star::
bigcpu::
```

Users can add or remove hosts from this list and enable or disable access control using the `xhost` command from the same machine as the server.

The X protocol intrinsically does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. X Servers that support the SECURITY extension fare better because clients can be designated untrusted via the authorization they use to connect; see the `xauth(1)` manual page for details. Restrictions are imposed on untrusted clients that curtail the mischief they can do. See the SECURITY extension specification for a complete list of these restrictions.

Sites that have better authentication and authorization systems might wish to make use of the hooks in the libraries and the server to provide additional security models.

signals

This entire section comes from `XFree86 Xserver(1)` manual page.

The X Server attaches special meaning to the signals in the following table.

Signals

Signal	Description
SIGHUP	Causes the server to close all existing connections, free all resources, and restore all defaults. It is sent by the display manager whenever the main user's main application (usually an xterm or window manager) exits to force the server to clean up and prepare for the next user.
SIGTERM	Causes the server to exit cleanly.
SIGUSR1	Used quite differently from either of the above. When the server starts, it checks to see if it has inherited SIGUSR1 as SIG_IGN instead of the usual SIG_DFL. In this case, the server sends a SIGUSR1 to its parent process after it has set up the various connection schemes. Xdm uses this feature to recognize when connecting to the server is possible.

starting the X Server from the command line

Starting X from the command line is not the preferred method of starting X on HP-UX. This documentation outlines the necessary steps to start X from the command line and the command line options for those users who may need to do so. The documentation provided here is based on the `Xf86(1)` manual page.

Itanium systems support two distinct X Servers. The display server is based on XFree86 and is in `/usr/bin/X11/Xf86`. The Technical Print Server (TPS) is based on HP's X Server and is in `/usr/bin/X11/Xhp`. The desktop wants to invoke the X Server as `/usr/bin/X11/X`. To solve the problem of needing to be able to invoke either of the X Servers, an X loader has been added in `/usr/bin/X11/X`. The X loader is responsible for deciding which X server to invoke. When the X loader starts one of the X Servers, it passes all the arguments to the X Server.

The following example demonstrates how to start the Xserver from the command line.

```
/usr/bin/X11/Xf86 [options …]
```

The command line options are specified in the following table.

X Server Command Line Options

Switch	Value	Description
:	DsplyNum	The X Server runs as the given **DsplyNum**. If multiple X Servers are to run simultaneously on a host, each must have a unique display number. The default value is 0.
–a	Num	Sets the pointer acceleration.
–ac	NA	Disables host-based access control mechanisms. Enables access by any host, and permits any host to modify the access control list. Use with caution. This option exists primarily for running test suites remotely.
–allowMouseOpenFail	NA	Allows the server to start up even if the mouse device can't be opened or initialized. This is equivalent to the **AllowMouseOpenFail** XF86Config file option.
allowNonLocalXvidtune	NA	Allows xvidtune to be run as a non-local client.
–ar1	NA	Sets XKB autorepeat delay.
–ar2	NA	Sets XKB autorepeat interval.
–audit	Level	Sets the audit trail level. The default level is 1, meaning only connection rejections are reported. Level 2 additionally reports all successful connections and disconnects. Level 4 enables messages from the SECURITY extension, if present, including generation and revocation of authorizations and violations of the security policy. Level 0 turns off the audit trail. Audit lines are sent as standard error output.
–auth	AutFile	Specifies a file which contains a collection of authorization records used to authenticate access.
–bestRefresh	NA	Chooses modes with the best refresh rate.
–broadcast	NA	Enables XDMCP and broadcast BroadcastQuery packets to the network. The first responding display manager is chosen for the session.
bc	NA	Enables bug compatibility.
+bs	NA	Enables any backing store support.
–bs	NA	Disables backing store support on all screens.
–c	NA	Turns off key-click.
c #	Volume	Sets key-click volume (allowable range: 0-100).

X Server Command Line Options (Continued)

Switch	Value	Description
–cc	Class	Sets the visual class for the root window of color screens. The class numbers are as specified in the X protocol.
–class	Class	XDMCP has an additional display qualifier used in resource lookup for display-specific options. This option sets that value, by default it is "MIT-Unspecified" (not a very useful value).
–co	FileName	Sets name of RGB color database. The default is: `/etc/X11/rgb.`
–core	NA	Causes the server to generate a core dump on fatal errors.
–cookie	Cookie	When testing XDM-AUTHENTICATION-1, a private key is shared between the server and the manager. This option sets the view of that private data (not that it is very private, being on the command line!).
disableVidMode	NA	Disables mode adjustments with xvidtune.
–displayID	DisplayID	Yet another XDMCP specific value, this one allows the display manager to identify each display so that it can locate the shared key.
–dpi int	Resolution	Sets the resolution of the screen, in dots per inch. To be used when the server cannot determine the screen size from the hardware.
dpms	NA	Enables VESA DPMS monitor control.
–dpms	NA	Disables VESA DPMS monitor control.
–deferglyphs	Fonts	Specifies the types of fonts for which the server should attempt to use deferred glyph loading. Fonts can be **all** (all fonts), **none** (no fonts), or **16** (16 bit fonts only).
–depth	Depth	Sets the default color depth. Legal values are 1, 4, 8, 15, 16, and 24. Not all drivers support all values.
–f #	Volume	Sets feep (bell) volume (allowable range: 0-100).
–fbbpp	Value	Set bpp for the frame buffer. Default: 8.
–fc	Font	Sets default cursor font.
–fn	FontName	Default font name.
–flipPixels	NA	Swaps the default values for the black and white pixels.
–from	LocalAddress	Specifies the local address to connect from.

X Server Command Line Options (Continued)

Switch	Value	Description
–fp	FontPath	Sets the search path for fonts. This path is a comma separated list of directories which the X Server searches for font databases.
–gamma –bgmma –ggamma –rgamma	Value Bvalue Gvalue Rvalue	Set the gamma correction. *value* must be between 0.1 and 10. The default is 1.0. This value is applied equally to the RGB values. The gamma values can be set independently with the -rgamma, -bgamma, and -ggamma options
–help	NA	Prints a usage message.
–l	NA	Causes all remaining command line arguments to be ignored.
–ignoreABI	NA	Makes module ABI mismatches non-fatal.
–indirect	HostName	Enables XDMCP and send IndirectQuery packets to the specified host.
–kb	NA	Disables the XKEYBOARD extension if present.
+kb	NA	Enable the X Keyboard Extension
–keyboard	KeyID	Uses the XF86Config file **InputDevice** section called KeyID as the core keyboard. By default the core keyboard input device referenced by the default **Layout** section are used, or the first relevant **InputDevice** section when there are no **Layout** sections.
–ld	INT	Limits data space to N Kb.
–lf	INT	Limits number of open files to N.
–ls	INT	Limits stack space to N Kb.
–nolock	NA	Disables the locking mechanism.
–logo	NA	Enables logo in screen saver.
nologo	NA	Disables logo in screen saver.
–layout	LayoutID	Uses the XF86Config file **Layout** section called LayoutID. By default the first **Layout** section is used.
–logfile	FileName	Uses the file called FileName as the X Server log file. The default log file is `/var/X11/Xserver/logs/Xf86.n.log`, where *n* is the display number of the X Server.

X Server Command Line Options (Continued)

Switch	Value	Description
–logverbose	Level	Sets the verbosity level for information printed to the X Server log file. When the *n* value is supplied, the log file verbosity level is set to that value. The default log file verbosity level is –1, which provides minimal output information.
–modulepath	Path	Sets the module search path to Path. Path is a comma separated list of directories to search for X Server.
–noloadxkb	NA	Prevents loading of the XKB keymap description.
–nolisten	TransType	Disables a transport type. For example, TCP/IP connections can be disabled with `–nolisten tcp`.
–norest	NA	Prevents a server reset when the last client connection is closed. This overrides a previous `–terminate` command line option.
–nosilk	NA	Disables Silken Mouse.
–once	NA	Causes the server to terminate (rather than reset) when the XDMCP session ends.
–p	Minutes	Sets screen–saver pattern cycle time in minutes.
–pixmap24	NA	Uses 24bpp pixmaps for depth 24.
–pixmap32	NA	Uses 32bpp pixmaps for depth 24.
–pn	NA	Permits the server to continue running if it fails to establish all of its well-known sockets (connection points for clients), but establishes at least one.
–nopn	NA	Rejects failure to listen on all ports.
–pointer	PointerID	Uses the XF86Config file **InputDevice** section called PointerID as the core pointer. By default the core pointer input device referenced by the default **Layout** section are used, or the first relevant **InputDevice** section when there are no **Layout** sections.
–port	PortNum	Uses an alternate port number for XDMCP packets. Must be specified before any –query, –broadcast or –indirect options.
probeonly	NA	Probes for devices, then exit.
–query	HostName	Enables XDMCP and send Query packets to the specified host.
–quiet	NA	Suppresses most informational messages at startup. The verbosity level is set to zero.

X Server Command Line Options (Continued)

Switch	Value	Description
–r	NA	Turns off auto-repeat.
r	NA	Turns on auto-repeat.
–s	Minutes	Sets screen-saver timeout time in minutes.
–scanpci	NA	When this option is specified, the X Server scans the PCI bus, and prints out some information about each device that was detected.
–screen	ScreenID	Uses the XF86Config file **Screen** section called **ScreenID**. By default the screens referenced by the default **Layout** section are used, or the first **Screen** section, when there aren't any **Layout** sections.
–su	NA	Disables save under support on all screens.
–sp	FileName	Causes the server to attempt to read and interpret FileName as a security policy file. The file is read at server startup and reread at each server reset. See the section about security on page 4-24 for more details.
–t	Number	Sets pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect).
–terminate	NA	Causes the server to terminate at server reset, instead of continuing to run. This overrides a previous `–noreset` command line option.
–to	Seconds	Sets default connection timeout in seconds.
–tst	NA	Disables all testing extensions (such as XTEST, XTrap, XTestExtension1, RECORD).
ttyxx	NA	Server started from init on `/dev/ttyxx`
v	NA	Sets video-off screen-saver preference.
–v	NA	Sets video-on screen-saver preference.
–verbose	Level	Sets the verbosity level for information printed on `stderr`. When the *n* value is supplied, the verbosity level is set to that value. The default verbosity level is –1.
–version	NA	Prints out the server version, patch level, release date, the operating system/platform it was built on, and whether it includes module loader support.
–weight	Weight	Sets RGB weighting at 16 bpp. The default is 565.

X Server Command Line Options (Continued)

Switch	Value	Description
–wm	NA	Forces the default backing-store of all windows to be **WhenMapped**. This is a backdoor way of getting backing-store to apply to all windows. Although all mapped windows have backing store, the backing store attribute value reported by the server for a window is the last value established by a client. If it has never been set by a client, the server reports the default value, **NotUseful**. This behavior is required by the X protocol, which allows the server to exceed the client's backing store expectations but does not provide a way to tell the client that it is doing so.
–x	Extension	Loads the specified extension at init.
–xf86config	FileName	Reads the server configuration from FileName.
+xinerama	NA	Enables XINERAMA extension.
–xinerama	NA	Disables XINERAMA extension.
–xkbcomp	NA	Default keymap compiler.
–xkbdb	NA	File that contains default XKB keymaps.
–xkbmap	NA	XKB keyboard description.

mapping options from the previous hp X Server to the current XFree86 X Server

The purpose of this section is to provide the user, who is familiar with the X* screens files for the HP X Server, a method of setting the equivalent options in the XF86Config file, in the current release of the XFree86 X Server. Only those options that are currently implemented in the release are documented here.

defaultVisual option

Class

The default class visual can be set in a Display subsection of the Screen section of the XF86Config file using the Visual option. The following example demonstrates how this would be done in the X*screens file and how it would be done in the XF86Config file. The example sets the default visual class to TrueColor.

```
X*screens File Example:

Screen /dev/crt
     DefaultVisual
         Class TrueColor

XF86Config File Example:

Section “Screen”
        . . .
     SubSection “Display”
        . . .
          Visual “TrueColor”
         . . .
     EndSubSection
        . . .
 EndSection
```

The default visual can be set to either PseudoColor or TrueColor.

Depth

The default depth of the visual can be set in the Screen section of the XF86Config file using the DefaultDepth option. The following example sets the default depth to 24.

```
X*screens File Example:

Screen /dev/crt
     DefaultVisual
          Depth 24

XF86Config File Example:

Section “Screen”
     . . .
     DefaultDepth 24
     . . .
EndSection
```

The depth may be selected as the default depth: 24.

minimum monitor power save level option

See display power management signaling (DPMS) on page 4-16 for more details on this option.

- **HPCursorScaleFactor <n>**

  See cursor scaling on page 4-19 for more details regarding this option.

- **No Server Logging**

  See logging and verbosity on page 4-19 for more details.

- **DisableGlxVisuals**

  This can be accomplished by not loading the GLX driver in the Modules section. See the example below.

  ```
  Section “Modules”
       Noload   “glx”
  EndSection
  ```

- **DPMSStandbyTime <Time (Seconds)>**
  DPMSSuspendTime <Time (Seconds)>
  DPMSOffTime <Time (Seconds)>

  See section on display power management signaling (DPMS) on page 4-16 for more details on these options.

- **HideDuplicateGlxVisuals**

  See display power management signaling (DPMS) on page 4-16 for more details on this option.

- **MakeDeviceGrayScale**

  This can effectively be done by selecting a “PseudoColor” visual and then setting the RGB values of each color entry identical, in order to create a gray scale colormap.

input devices

keyboards

supported keyboard drivers

The supported keyboard driver is:

```
keyboard
```

supported keyboard options

The following is a list of keyboard options supported by HP.

Supported Keyboard Options

Options	Value	Description
AutoRepeat	Integer	Set the keyboard auto repeat parameters. Not all platforms implement this.
Xleds	Integer …	Specify which keyboard LEDs can be user controlled (for example, with `xset(1)`).

pointers

supported pointer drivers

The supported pointer driver is:

```
mouse
```

supported pointer options

The following is a list of pointer options supported by HP.

Printer Options Supported by HP

Options	Value	Description
Protocol	String	Values may only be “PS/2”
Device	String	The value may only be “`/dev/hid/mouse_000`”

output devices

ATI Fire GL4™ device-dependent information

ATI Fire GL4 provides 8 overlay planes, 48 image planes, a 24-bit Z buffer, 4 8-bit per channel hardware colormaps and 1 10-bit per channel hardware colormap for use in gamma correction. This device provides 2D hardware acceleration for most operations as well as 3D acceleration for lighting, shading and texture mapping.

supported visuals

ATI Fire GL4 supports the following visuals:

Visuals Supported by ATI Fire GL4

Class	Depth	Layer
PseudoColor	8	Overlay
TrueColor	24	Image

Both the PseudoColor and TrueColor visuals are enabled by default. See supported device options on page 4-39 for instructions on changing the default visual and/or disabling overlay visuals.

supported device options

Supported Device Options

Options	Value	Default	Description
enableDVI	Boolean	False	Enable DVI connector.
Qbs	Boolean	False	Enable quad-buffered stereo mode.
FrameLock	Boolean	False	Synchronizes frame buffer refresh between multiple raster engines. Setting this option causes the device to become a slave to an external timing generator (another device). Commonly used to sync devices in a stereoscopic display environment.
CountTransInOvlyVis	Boolean	True	When set to false, causes the X Server to reserve the transparent pixel index in all PseudoColor overlay colormaps, thus reporting only 255 available entries.
TransparentIndex0	Boolean	False	Make the overlay transparent pixel index 0 instead of 255.
DefaultVisualTrueColor	Boolean	True	Use TrueColor as the default visual instead of PseudoColor.

Supported Device Options

Options	Value	Default	Description
Overlay	Boolean	True	Enable the Overlay visuals.
FSAA	Drop down	False	Enable Full Screen Anti Aliasing

These options are enabled by adding a line to the `/etc/X11/XF86Config` file in the Device section. For example:

```
Option "Overlay" "True"
```

Note that both the option name and option value must be enclosed in quotation marks.

supported monitor configurations

The following table documents supported display resolution and refresh rate for the ATI Fire GL4. Check your monitor specification to determine if the monitor supports any or all of these resolutions.

Supported Monitor Options

Resolution (HxV)	Frequency (Hz)	Description
1024x768	75	VESA Standard
1024x768	85	VESA Standard
1280x1024	60	VESA Standard
1280x1024	75	VESA Standard
1280x1024	85	VESA Standard
1600x1024	75	24" monitor
1600x1024	85	24″ monitor
1600x1200	75	VESA Standard
1600x1200	85	VESA Standard
1920x1080	75	24" monitor
1920x1080	85	24" monitor
1920x1200	75	24" monitor
1920x1200	85	24" monitor

ATI Fire GL4 configuration hints

overlay visuals and overlay transparency

ATI Fire GL4 devices have one visual in the overlay planes, depth-8 PseudoColor. To allow applications to determine which visuals are in the overlay planes, overlay visuals are listed in the "SERVER_OVERLAY_VISUALS" property attached to the root window. The default overlay visual has a transparent type of "1" (TransparentPixel).

If you need an overlay colormap that supports transparency, create the colormap using the visual that has transparency in its SERVER_OVERLAY_VISUALS property. To look at the contents of this property, you would use code similar to the following:

```
{
    typedef struct {
            VisualID    overlayVisualID;
            Card32       transparentType;
            /* None, TransparentPixel, TransparentMask */
            Card32      value;
            /* Either pixel value or pixel mask */
            Card32      layer;
    } OverlayVisualPropertyRec;
    OverlayVisualPropertyRec  *pOverlayVisuals, *pOVis;
    XVisualInfo               getVis;
    XVisualInfo               *pVisuals;
    Atom                      overlayVisualsAtom, actualType;
    ...
    /* Get the visuals for this screen and allocate. */
    getVis.screen   = screen;
    pVisuals       = XGetVisualInfo(display, VisualScreenMask,
                                    &getVis, &nVisuals);
    pOverlayVisuals = (OverlayVisualPropertyRec *)
                      malloc ( (size_t)nVisuals *
                               sizeof(OverlayVisualPropertyRec)

                           );

    /*
    ** Get the overlay visual information for this screen.  Obtain
    ** this information from the SERVER_OVERLAY_VISUALS property.
    */
    overlayVisualsAtom = XInternAtom(display,"SERVER_OVERLAY_VISUALS",
                                     True);
    if (overlayVisualsAtom != None)
    {
    /* Since the Atom exists, request the property's contents. */
     bytesAfter = 0;
     numLongs = (nVisuals * sizeof(OverlayVisualPropertyRec) + 3 ) / 4;

   XGetWindowProperty(display, RootWindow(display, screen),
                      overlayVisualsAtom, 0, numLongs, False,
                      AnyPropertyType, &actualType, &actualFormat,
                      &numLongs, &bytesAfter, &pOverlayVisuals);
   if (bytesAfter != 0 ) {/* Serious Failure Here */} ;

   /* Loop through the pOverlayVisuals array. */
           ...
   nOVisuals = numLongs/sizeof(OverlayVisualPropertyRec);
```

```
    pOVis     = pOverlayVisuals;

    while (--nOVisuals >= 0)
                {if ( pOVis->transparentType == TransparentPixel )
                     {/*
                     **Found a transparent overlay visual,
                     **set ident. aside.
                     */
                };
                pOVis++;
    }
    XFree(pOverlayVisuals);
    /*
    **There might be some additional checking of the found
    **transparent overlay visuals wanted; e.g., for depth.
    */
    }
    XFree(pVisuals);
}
```

This program segment is not complete; however, its main purpose is to give an idea of how to find an overlay visual having transparency.

When the overlay planes are enabled, one colormap entry in the PseudoColor colormaps is not available for use by clients. The server handles this entry in one of two ways depending upon the setting of the "CountTransInOvlyVis" device option. If the option is not set, the server reports that 256 colormap entries are available for allocation in the PseudoColor visual. This option may be useful to applications that depend on having 256 colormap entries available in depth 8 PseudoColor visuals. However, the transparent pixel always remains transparent. The image layer is visible wherever this pixel value is rendered in the overlay planes. Hence, applications should not render using this pixel unless transparency is desired.

This may cause problems with some applications. Setting "CountTransInOvlyVis" causes the server to reserve the transparent pixel index. In this case, the server reports that 255 colormap entries are available for allocation in the PseudoColor visual. The transparent index value is configurable through the "TransparentIndex0" device option. Setting this option changes the transparent pixel value to 0 from the default of 255. This option may be useful to applications that depend on the transparent pixel value being 0.

ATI Fire GL4 Colormaps Hints

ATI Fire GL4 devices have a total of 5 hardware colormaps. One colormap is reserved for gamma correction and is not directly available to X clients. The remaining colormaps are available for clients to use. When the default visual is in the overlay planes, one of the four available hardware colormaps is reserved for the default colormap. This ensures that clients using the default colormap never encounter color flashing.

The ATI Fire GL4 driver reports on the installation status of a total of 5 colormaps via the XListInstalledColormaps() API. Four of these correspond to the actual state of the hardware color tables used with PseudoColor visuals. Colormaps are installed in these hardware LUTs in first-in, first-out order. The driver also keeps track of the last TrueColor colormap installed via XInstallColormap() and includes this information whenever queried. By definition, all TrueColor colormaps are always installed.

ATI FireGL X1and Z1 device-dependent information

ATI FireGL X1 and ATI FireGL Z1 provide 8 overlay planes, 48 image planes, a 24-bit Z buffer, 4 8-bit per channel hardware colormaps and 1 10-bit per channel hardware colormap for use in gamma correction. These devices provide 2D hardware acceleration for most operations as well as 3D acceleration for lighting, shading and texture mapping.

supported visuals

ATI FireGL X1 and ATI FireGL Z1 supports the following visuals:

Visuals Supported by ATI FireGL X1 and Z1

Class	Depth	Layer
PseudoColor	8	Overlay
TrueColor	24	Image

Both the PseudoColor and TrueColor visuals are enabled by default. See supported device options on page 4-39 for instructions on changing the default visual and/or disabling overlay visuals.

supported device options

Supported Device Options

Options	Value	Default	Description
Qbs	Boolean	False	Enable quad-buffered stereo mode.
DefaultVisualTrueColor	Boolean	False	Use TrueColor as the default visual instead of PseudoColor.
Overlay	Boolean	True	Enable the Overlay visuals.
EnableOpaqeOvelayVisual	Boolean	False	Enable the Opaque Overlay visual.

These options are enabled by adding a line to the `/etc/X11/XF86Config` file in the Device section. For example:

```
Option "Overlay" "True"
```

Note that both the option name and option value must be enclosed in quotation marks.

supported monitor configurations

The following table documents supported display resolution and refresh rate for the ATI FireGL X1 and ATI FireGL Z1. Check your monitor specification to determine if the monitor supports any or all of these resolutions.

Supported Monitor Options

Resolution (HxV)	Frequency (Hz)	Description
1024x768	75	VESA Standard
1024x768	85	VESA Standard
1280x1024	60	VESA Standard
1280x1024	75	VESA Standard
1280x1024	85	VESA Standard
1600x1024	75	24" monitor
1600x1024	85	24″ monitor
1600x1200	75	VESA Standard
1600x1200	85	VESA Standard
1920x1080	75	24" monitor
1920x1080	85	24" monitor
1920x1200	75	24" monitor
1920x1200	85	24" monitor

ATI Radeon™ 7000, Manager Processor and rx5670 graphics solution device-dependent information

supported visuals

There are two visuals available with the ATI Radeon7000 display, although not at the same time. The X server can be either in depth 8 or depth 24 frame buffer mode. In depth 24 mode the available visual is TrueColor. This is the recommended visual. In depth 8 mode the available visual is PseudoColor. The frame buffer depth is configurable through HP's System Administration Manager (SAM).

Visuals Supported by ATI Radeon7000

Class	Depth	Layer
PseudoColor	8	Overlay
TrueColor	24	Image

supported device options

Supported Device Options

Options	Value	Default	Description
NoAccel	Boolean	False	Turn off hardware acceleration
SWcursor	Boolean	False	Use software cursor instead of hardware
crt_screen	Boolean	False	Use only crt, no digital flat panel
PanelSize	Integer	None	Flat panel size override
UseFBDev	Boolean	False	Use simple frame buffer device driver

supported monitor configurations

The following table documents supported display resolution and refresh rate for the ATI Radeon7000. Check the monitor's specification to determine which resolutions and frequencies are supported by the monitor.

Supported Monitor Configurations

Resolution (HxV)	Frequency (Hz)	Description
1024x768	75	VESA standard
1024x768	85	VESA standard
1280x1024	60	VESA standard, Flat Panel
1280x1024	75	VESA standard
1280x1024	85	VESA standard

Supported Monitor Configurations

Resolution (HxV)	Frequency (Hz)	Description
1600x1024	75	24" monitor
1600x1024	85	24" monitor
1600x1200	75	VESA standard
1600x1200	85	VESA standard
1920x1080	75	24" monitor
1920x1080	85	24" monitor
1920x1200	75	24" monitor

ATI Radeon7000 configuration hints

Depth 24 is the recommended depth. The benefit is that with the TrueColor visual applications have a large 16M color palette available. Because the TrueColor colormap is read only, it can be shared among multiple applications without any color flashing problems.

The depth 8 frame buffer configuration is mainly intended for backwards compatibility. Some older applications, or applications that want to use color map tricks may insist on using the PseudoColor visual. The PseudoColor colormap is read/write and has 256 colors. Each application can have its own private colormap if it asks for it, or they can share the same default colormap.

An issue with the PseudoColor visual is that there is only one hardware colormap. If there are multiple applications running, each with its own private logical colormap, they have to share the single hardware colormap. Only one application, the one that has the focus, has the correct colors. Others do not. You may observe color “flashing” when you move focus between different windows. If PseudoColor applications can share the same default colormap without running out of colors, the color flashing does not occur.

system requirements

hardware compatibility table

This table lists graphics cards that are support on a system and which OS is required.

Hardware Compatibility Table

Graphics Device	Supported Systems	Required Operating System
ATI Fire GL4	zx2000, zx6000	HP-UX 11.22
ATI Radeon7000	zx2000, zx6000	HP-UX 11.22
Manager Processor	zx6000	HP-UX 11.22
rx5670 Graphics Solution	rx5670	HP-UX 11.22
ATI FireGL X1 and Z1	zx2000, zx6000	HP-UX 11.23

monitor compatibility

Most multi-sync monitors are compatible with the graphics cards in this release of the X Server. There are two statements that can be made regarding monitors:

- The graphics cards described in this document work with any monitor with 1280x1024 @ 75 Hz. or 1024x768@75 Hz. resolutions.
- If the selected monitor resolution and frequency worked with any HP Visualize graphics card (fxe, fx5 or fx10), then the graphics cards documented here also work with a monitor of the same resolution and frequency.

compatibility matrix with previous releases

The following table illustrates differences between the graphics devices of previous HP releases and the graphics device of this release.

Compatibility Matrix

Feature	Visualize fxe	Visualize fx5/fx10	ATI Fire GL4	ATI Radeon7000	ATI FireGL X1/Z1
Overlay Planes	8	8	8/8	8/8	8/8
Overlay LUTs	2	2	4	4	Infinite
Image Planes	32	32	32	32	32
Image LUTs	2	2	1	4	1
Per pixel Attributes	Yes	Yes	Yes	Yes	No
Default Visual	8I-O	8I-O	24RGB-I	24RGB-I	8I-O
Overlay Transparency	Yes	Yes	Yes	Yes	Yes
Gamma Correction	Yes	Yes	Yes	Yes	Yes
Buffer swap method	Register write	Register write	Register write	Register write	Copy swap
Stereo support	No	Yes (external sync and glasses)	Yes (glasses)	Yes (glasses)	Yes (Dec. 2003)
Clip rectangles	4	4	1	1	1
Clip plane	Yes	Yes	Yes	Yes	Yes
Hardware byte swapping	Yes	Yes	Yes	Yes	Yes
Video conversion	No	Yes	Yes	Yes	Yes
1920x1200 @ 76 Hz	No	Yes	Yes	Yes	Yes
1600x12 @ 85 Hz	Yes	Yes	Yes	Yes	Yes

Compatibility Matrix

Feature	Visualize fxe	Visualize fx5/fx10	ATI Fire GL4	ATI Radeon7000	ATI FireGL X1/Z1
1280x1024 @ 85 Hz	Yes	Yes	Yes	Yes	Yes
1024x768 @ 85 Hz	Yes	Yes	Yes	Yes	Yes
Hardware multi display	No	No	Yes (ATI Fire GL4 3:3200x1200)	Yes	Yes

miscellaneous

fonts

The X Server can obtain fonts from directories or font servers. Setting up a font server or making a directory a font directory is beyond the scope of this document. The font path can be loaded via the –fp option from the command line or from the XF86Config file. The latter is the preferred method. The default font path is: `/usr/lib/X11/fonts/misc`. See the Files section on page 4-6 regarding the **FontPath**. The following font directories are delivered with the system and may be added to the font path. Applications may install their own fonts. The application font path can be added to the FontPath as necessary.

```
/usr/lib/X11/fonts/misc/
/usr/lib/X11/fonts/hp_kana8/
/usr/lib/X11/fonts/hp_roman8/75dpi/
/usr/lib/X11/fonts/iso_8859.1/100dpi/
/usr/lib/X11/fonts/iso_8859.1/75dpi/
/usr/lib/X11/fonts/hp_chinese_s/75dpi/
/usr/lib/X11/fonts/hp_chinese_t/75dpi/
/usr/lib/X11/fonts/hp_korean/75dpi/
/usr/lib/X11/fonts/hp_japanese/100dpi/
/usr/lib/X11/fonts/iso_8859.2/75dpi/
/usr/lib/X11/fonts/iso_8859.5/75dpi/
/usr/lib/X11/fonts/iso_8859.6/75dpi/
/usr/lib/X11/fonts/iso_8859.7/75dpi/
/usr/lib/X11/fonts/iso_8859.8/75dpi/
/usr/lib/X11/fonts/iso_8859.9/75dpi/
/usr/lib/X11/fonts/iso_8859.15/75dpi/
```

files

The X Server makes use of various files on the system during normal operation. The section lists the default location of the files and gives a brief description of what they do.

Default Locations of Configuration Files

File	Description
/etc/X11/XF86Config	The configuration file. The X Server uses this file to configure itself during initialization.
/etc/X11/rgb	The color database.
/etc/Xn.hosts	Initial access control list for display n.
/var/X11/XServer/logs/Xf86.n.log	The log file, where n is the display number.

5

X Server configuration details

This chapter discusses several details concerning the configuration of X hosts, colormaps, mouse and keyboard.

making an x*.hosts file

The `/etc/X0.hosts` file is an ASCII text file containing the host names of each remote host permitted to access your local server.

- If you are running as a stand-alone system, you must have your system's name in this file.
- If you are part of a network, the other system names must be included.

The syntax is as follows:

<*host*>
<*host*>
<*host*>

For example, if you are hpaaaaa, and regularly ran clients on hpccccc, and hpddddd, you would want the following lines.

```
hpaaaaa
hpccccc
hpddddd
```

Note that aliases work as well as host names, provided they are valid, commonly known across the network.

using an /etc/hosts file

This file need not be present if your system is configured to query a nameserver.

The `/etc/hosts` file is an ASCII text file containing a list of all the host names and internet addresses known to your system, including your own system.

If your system is not connected to a network, use the loopback address (127.0.0.1) and the hostname unknown:

127.0.0.1 unknown

For a local system to access a remote host:

- The address and hostname of the remote host must be listed in the local system's `/etc/hosts` file.
- The user must have a valid login (username and password) and home directory on the remote host.

initializing the colormap with xintcolormap

The `xinitcolormap` client initializes the X colormap. Specific X colormap entries (pixel values) are made to correspond to specified colors. An initialized colormap is required by applications that assume a predefined colormap.

`xinitcolormap` has the following syntax: `xinitcolormap` [<*options*>]

where the < *options*> are:

- `-f` <*colormapfile*>

  Specifies a file containing a colormap.

- `-display` <*display*>

  Specifies the server to connect to.

- `-c` <*count*>

  Only the first count colors from the colormap file will be used if this parameter is specified.

- `-k` or `-kill`

  Deallocate any colormap entries that were allocated by a previous run of `xinitcolormap`.

`xinitcolormap` chooses a colormap file in the order shown below. Once one is found, the other sources aren't searched.

1. The command line option [-f< *colormapfile*>].
2. Colormap default value.
3. The xcolormap file in `/usr/lib/X11`.
4. If no colormap file is found, this default colormap specification is assumed black (colormap entry 0), white, red, yellow, green, cyan, blue, magenta (colormap entry 7).

`xinitcolormap` should be the first client program run at the start of a session in order to assure that colormap entries have the color associations specified in the colormap file. Sometimes you may encounter this X toolkit warning:

```
X Toolkit Warning: cannot allocate colormap entry for 94c4d0
```

where "94c4d0" is a color specified in the application running. If this occurs, it means that you have probably reached the limit of colors for your graphics card/display combination. Executing xinitcolormap may solve the problem.

For more information about xinitcolormap, refer to its reference page.

6

miscellaneous topics

This chapter covers miscellaneous graphics issues relating to HP-UX 11.X.

thread-safing

Hewlett-Packard's OpenGL libraries are supported in multi-threaded applications (using POSIX threads). This documentation is not a tutorial on threads programming or multiprocessing application issues. For more, and general, information about the use of POSIX threads, consult the HP-UX documentation set.

One of the restrictions in using these APIs in multi-threaded programs is that The 3D graphics libraries support kernel threads only (libpthread); they do not support the DCE user threads package (libcma).

reference documentation

You may find the following documentation helpful when using HP graphics products:

For installing products:

- *HP-UX Reference*
- *System Administration Tasks*
- *Installing and Updating HP-UX*